Section 5. Process Control Subsystem

5.1 Introduction

The Process Control Subsystem (PCS) controls LPGS production planning and processing. It receives product generation requests from the ECS and sets up, monitors the status of, and controls processing of LPGS work orders. PCS monitors LPGS resources and provides processing status in response to customer and operator requests.

5.2 Design Overview

This section provides an overview of the PCS software design. It presents the relationships between the PCS and the other LPGS subsystems. It also discusses the assumptions, constraints, and considerations used in the design process.

5.2.1 Subsystem Software Overview

Figure 5–1 contains the PCS context diagram.

PCS receives Level 1 product generation requests from ECS via DMS and stores them in the database. PCS initiates processing of the product generation request by creating a work order. It selects the appropriate work order procedure to be run from the set of available procedures maintained in the database. The user interface (UI) populates the database with the list of procedures. The work order procedure identifies an ordered list of scripts that need to be run to produce the requested product. The procedure also supplies the default parameters for each script. PCS uses the options specified in the product generation request to override the default parameters. PCS stores the work order in the database and sets its state so that DMS is informed that the work order is ready for L0R ingest.

PCS starts work order processing when the input files have been successfully ingested and the system resources are available for processing. The work order specifies the order in which the scripts are to be run. The work order also indicates when processing is to be halted after a script completes; this allows for human intervention, if necessary. The scripts run DMS, QAS, RPS, and GPS application programs. PCS initiates execution of the work order scripts in sequence until the work order completes or it encounters a halt condition. PCS provides required parameters to each script. If work order processing completes successfully, PCS sends a message notifying DMS that the product is ready for transfer to ECS. In the case of a failure, PCS notifies the AAS analyst of the problem by recording the anomaly in the database. For a work order that has been halted, PCS waits for the operator to resume it using a UI tool. PCS periodically polls the database looking for the operator’s response. PCS resumes work order processing with the next script when system resources are available.

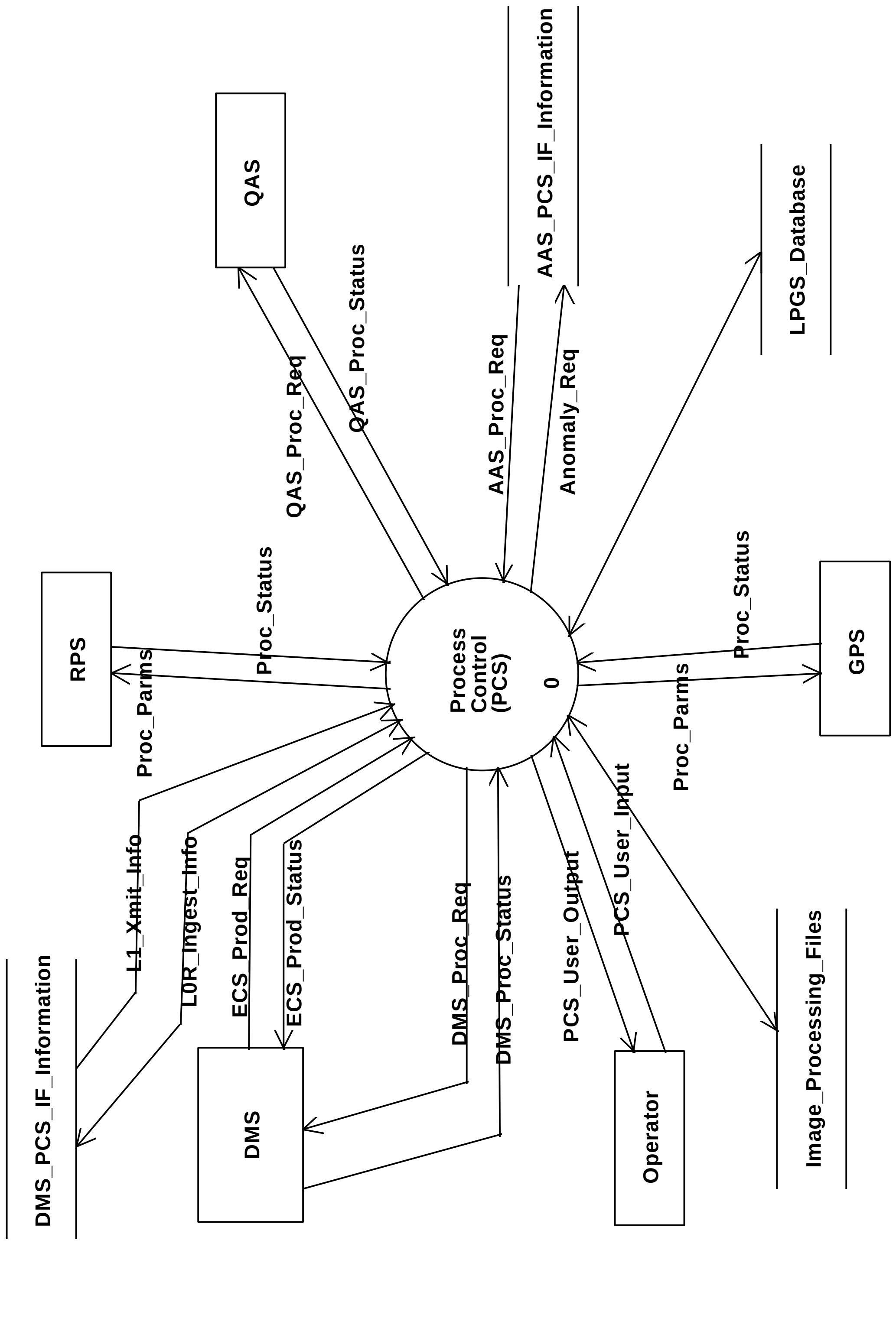

Figure 5–1. PCS Context Diagram

PCS also receives Level 1 product cancellation requests from ECS via DMS. After receiving confirmation from the operator that the request should be canceled (if confirmation is necessary), PCS terminates processing of the corresponding work order at a convenient point (e.g., between scripts). It updates the database to indicate that the work order and the product generation request have been canceled, and sends a message to ECS via DMS informing it of the cancellation. PCS also sends a message to ECS via DMS when it is unable to cancel the request.

5.2.2 Design Considerations

The LPGS PCS design approach was to reuse the IAS PCS design where the functions are similar.

5.2.2.1 Design Assumptions

The following assumptions were made in designing the PCS.

A. Order of Processing

PCS processes work orders on a first-in, first-out (FIFO) basis except when the operator explicitly changes the priority of a work order. When the required system resources are available, it selects the oldest work order among those with the highest priority to process next.

B. Parallel Processing of Work Orders

PCS supports concurrent processing of work orders. Operator-controlled parameters are used to restrict the number of in-progress and actively executing work orders. The parameters are used to control the flow through the production pipeline.

C. Cancellation Confirmation Option

An operator-controlled parameter specifies whether or not the LPGS operator must confirm product cancellation requests received from the ECS. If operator confirmation is required, PCS notifies the operator of each cancellation request. PCS continues to process “cancellation pending” work orders until a response is received from the operator.

D. Cancellation Points

PCS can cancel a product generation request at various points between receipt of the request and the completion of work order processing. Once DMS has been notified that the product is ready for transfer, the product generation request cannot be canceled. If the work order is currently executing, PCS waits for the script to terminate before processing the cancellation.

E. Visual Quality Assessment

PCS supports visual quality assessment through the use of halts. The QAS analyst performs the visual assessment and then resumes the work order using a UI tool. An operator-controlled parameter specifies whether visual quality assessments are to be performed.

5.3 Subsystem Design

This subsection provides a description of the PCS software architecture selected to implement the PCS design. Figure 5–2 shows the PCS subsystem software architecture. The PCS is composed of the Request Processor (PRQ), the Work Order Generator (PWG), the Work Order Scheduler (PWS), and the Work Order Controller (PWC) tasks. The PRQ and PWS tasks run in the background at all times. The LPGS initialization task starts and monitors all LPGS background tasks.

The PRQ task is responsible for receiving from DMS messages that are being forwarded from ECS. When it receives a product generation request, it performs a fork and exec of the Work Order Generator task to generate the work order for that request. The PRQ task may have multiple PWG processes running concurrently.

The PWS task is responsible for polling the database to determine when to start a work order. PWS nominally processes work orders in the order that they were received. A UI tool allows the operator to change the priority of a work order so that it can be moved ahead in the schedule. To initiate work order processing, the PWS task performs a fork and exec of the PWC task. The PWS uses a system level parameter to control the number of PWC tasks active at one time. The PWC task builds the environment of each script and performs a fork and exec of the script. A script runs one or more DMS, RPS, GPS, or QAS programs. PWC waits for script completion. It continues processing scripts until it reaches the last script or encounters a halt condition.

5.3.1 Request Processor (PRQ) Task

This subsection describes the PRQ task software.

5.3.1.1 Task Overview

The Request Processor task is primarily responsible for receiving requests (via the DMS) for product generation, production status, and product cancellation. It records the product generation and the product cancellation requests in the database. It starts the PWG task when it receives a new product generation request. For product status requests, PRQ retrieves status information from the database and sends the response to DMS (for forwarding to ECS).

5.3.1.2 Initialization

The PRQ task initialization consists of connecting to the database, establishing a link for intertask communications, and retrieving any setup parameters.

5.3.1.3 Normal Operation

During normal operations, the PRQ task performs in a loop until it receives a termination message. It calls xxx_get_msg to get the next message to be processed. Depending on the type of message received, it calls prq_proc_l1_gen_prod_req, prq_proc_l1_prod_stat_req, prq_proc_l1_cancel_req, prq_proc_l1_cancel_ack, prq_proc_cancel_confirm, or prq_term.

REVIEW

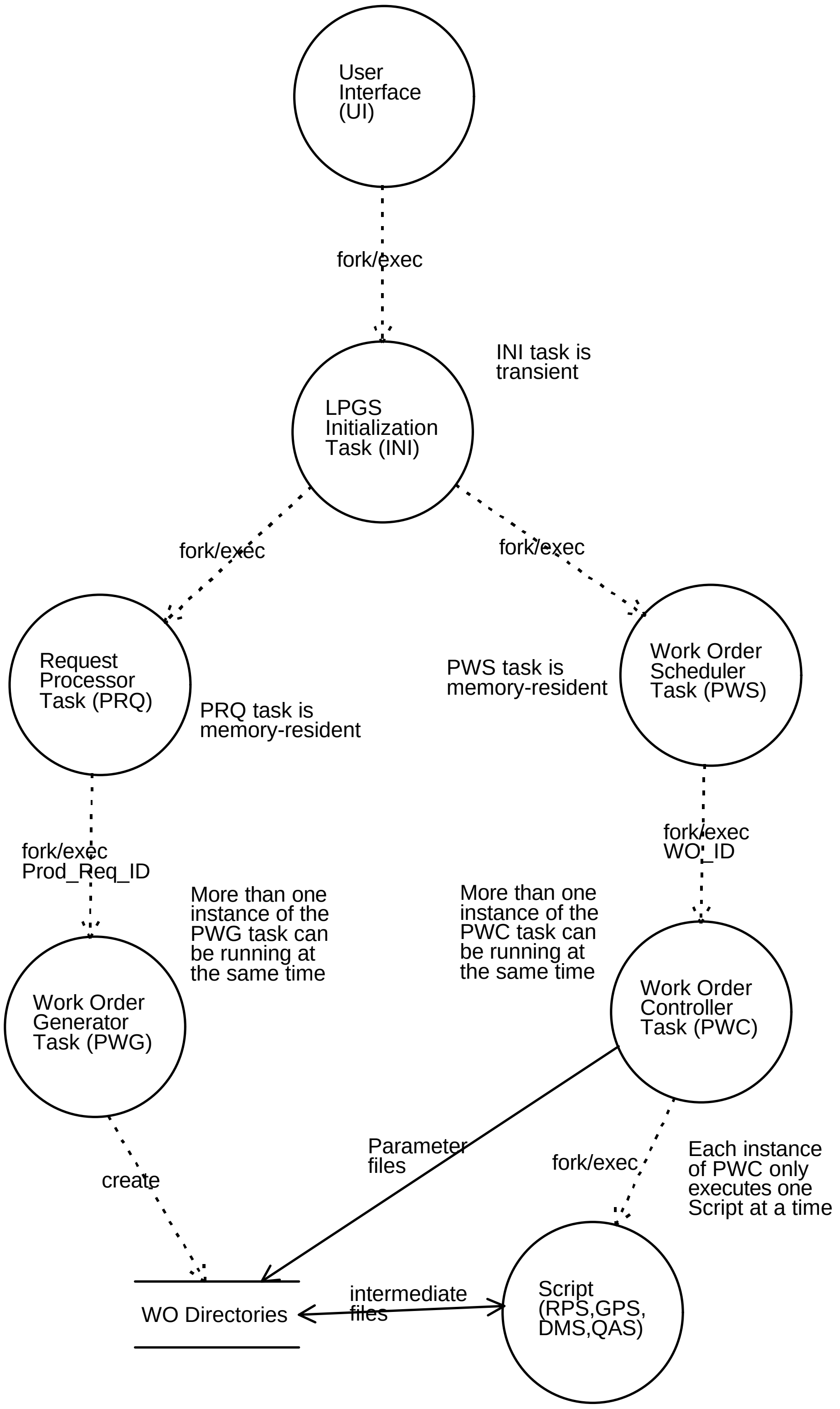
User Interface (UI)
fork/exec
INI task is transient
LPGS Initialization Task (INI)
fork/exec
fork/exec
Request Processor Task (PRQ)
PRQ task is memory-resident
PWS task is memory-resident
Work Order Scheduler Task (PWS)
fork/exec Prod_Req_ID
More than one instance of the PWG task can be running at the same time
More than one instance of the PWC task can be running at the same time
fork/exec WO_ID
Work Order Generator Task (PWG)
Work Order Controller Task (PWC)
Parameter files
create
fork/exec
Each instance of PWC only executes one Script at a time
WO Directories
intermediate files
Script (RPS,GPS, DMS,QAS)

Figure 5–2. PCS Subsystem Architecture

5.3.1.4 Error Handling

The Request Processor reports any errors encountered and exits gracefully when it is unable to continue processing.

5.3.1.5 Design

This subsection presents the design of the PRQ task. Figure 5–3 shows the structure chart for the Request Processor task. The module specifications for the PRQ task are provided below.

NAME: prq_main

TITLE: PRQ Main

BODY: This module processes messages that either originate at the ECS or within the PCS subsystem. The messages are as follows:

- Generate L1 product request
- L1 product status request
- L1 product cancel request
- L1 product cancel acknowledgment
- Cancellation confirmation
- Shutdown

This module runs in a loop until it receives a message to shutdown. It first calls the xxx_get_msg to get the next message to be processed. Depending on which message is received, it calls one of the following modules:

- prq_proc_l1_gen_prod_req
- prq_proc_l1_prod_stat_req
- prq_proc_l1_prod_cancel_req
- prq_proc_l1_prod_cancel_ack
- prq_proc_cancel_confim
- prq_term

NAME: prq_init

TITLE: PRQ Initialization

BODY: This module initializes the Request Processor task by connecting to the database and setting up communications for receiving and sending messages. It returns the prq_stat to show successful or unsuccessful completion.

NAME: prq_proc_l1_gen_prod_req

TITLE: Process Product Generation Request

BODY: This module receives the prq_l1_gen_prod_req that originated at the ECS. Data from the request is inserted into the L1_Prod_Requests table, and the Work Order Generator task is started to generate a work order for this request. The procedure returns the prq_stat to indicate successful or unsuccessful completion.

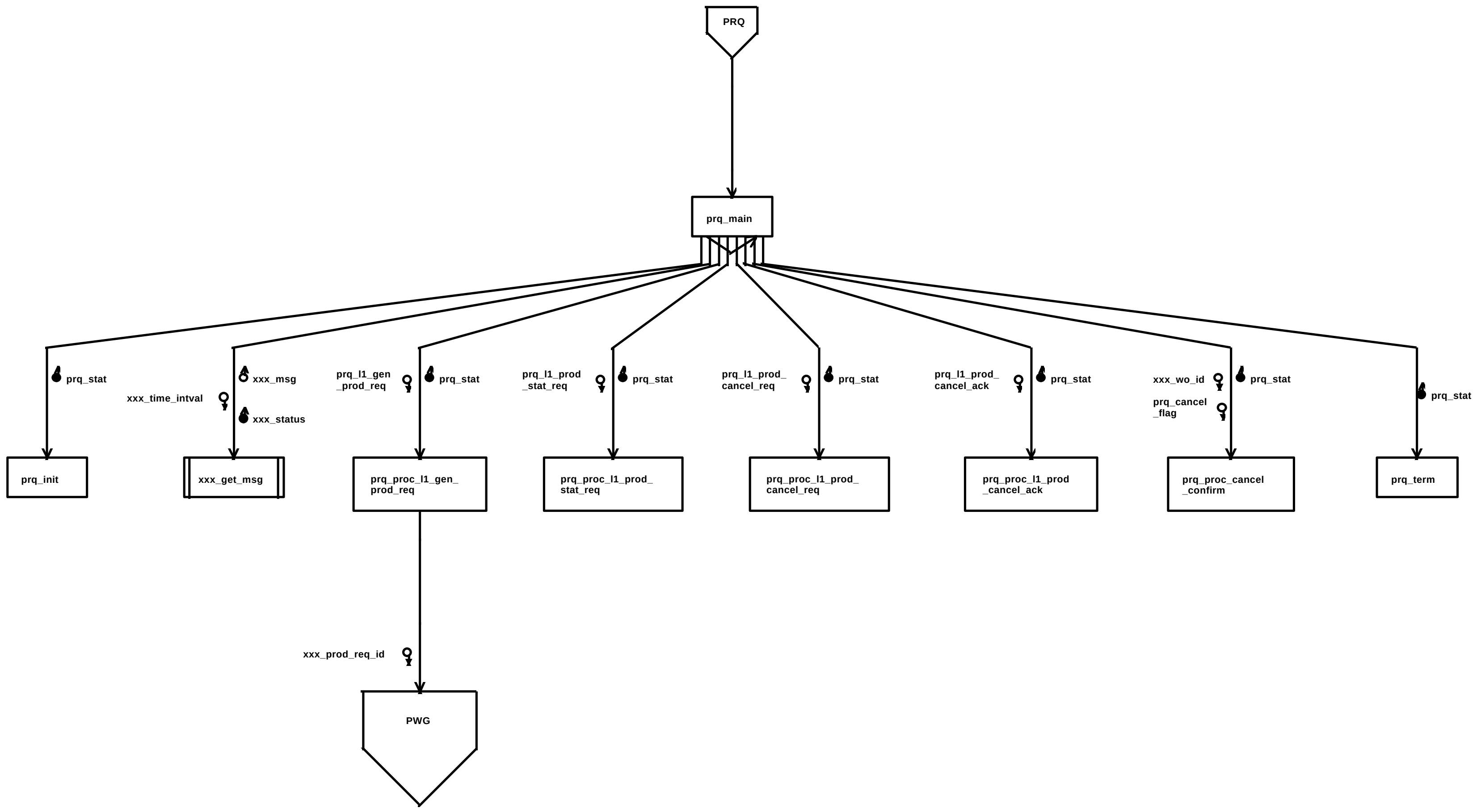

Figure 5–3. PRQ Task Structure Chart

NAME: prq_proc_l1_prod_stat_req

TITLE: Process Product Status Request

BODY: This module receives the prq_l1_prod_stat_req that originated at the ECS. The module checks the L1_Prod_Requests table for the status, generates a message containing the status, and sends the message to DMS to be sent to ECS. The prq_stat is returned to indicate successful or unsuccessful completion.

NAME: prq_proc_l1_prod_cancel_req

TITLE: Process Product Cancellation Request

BODY: This module receives prq_l1_prod_cancel_req to cancel a product generation request originating from the ECS. The module first checks the status of the request in the L1_Prod_Requests table. If the request has already been processed (but not delivered), delivered, failed, or canceled, then cancellation is not possible and the L1_Prod_Cancel_Stat is returned to the ECS indicating that the cancellation is not possible.

If the L1 request has not yet completed, then the cancellation confirmation flag is checked to see whether cancellation confirmation is required from the operator. If confirmation is required, then the cancellation status in the L1_ Prod_Requests is updated and a message is generated and sent to the operator.

If the L1 processing of the product request has not yet started, then the L1_Prod_Requests table is updated to mark the request as “canceled,” and the L1_Prod_Cancel_Stat is generated indicating that the cancellation has been performed. The L1_Prod_Cancel_Stat is sent to the ECS via DMS.

If the product generation request is running when the cancellation request is received, the cancellation status in the L1_Prod_Requests table is updated. Then, the Work Order Scheduler or Work Order Controller tasks can perform the cancellation when the product request is no longer running, i.e., when it has completed or encountered a halt.

If the product generation request is in a “halted” type of condition, the work order is marked canceled, the work order directories are marked for deletion, and the L1_Prod_Cancel_Stat is generated and sent to the ECS via DMS.

The prq_stat is returned to indicate successful or unsuccessful completion of the module.

NAME: prq_proc_l1_prod_cancel_ack

TITLE: Process Cancel Acknowledgement

BODY: This module processes the prq_l1_prod_cancel_ack, product generation cancellation acknowledgement, sent from the Work Order Scheduler or Work Order Controller task. When the acknowledgement is received, this module updates the L1_Prod_Requests table to indicate that the request has been canceled and sends the L1_Prod_Cancel_Stat to the ECS via DMS.

The prq_stat is sent to indicate successful or unsuccessful completion.

NAME: prq_proc_cancel_confirm

TITLE: Process Cancellation Confirmation

BODY: This module receives the prq_cancel_flag from the operator along with the xxx_wo_id. The operator can either approve or disapprove the ECS request to cancel the product generation request. If the operator indicates approval to continue with the cancellation, this procedure checks the status of the product request and work order using the xxx_wo_id. If the request has not yet begun processing, the L1_Prod_Cancel_Stat is generated and sent to the ECS and the cancellation status in the L1_Prod_Requests table is updated.

If the product request has already been completed, delivered, failed, or canceled, the L1_Prod_Requests table is updated to show the receipt of the request, and the L1_Prod_Cancel_Stat is generated and sent to the ECS via DMS. However, this L1_Prod_Cancel_Stat indicates that it is too late to cancel the product generation request. The cancellation status in the L1_Prod_Requests is updated.

If the product generation request is running, then the cancellation status in the L1_Prod_Request table is marked “confirmed.”

If the product generation request is in a “halted” type of condition, the work order is marked canceled, the work order directories are marked for deletion, and the L1_Prod_Cancel_Stat is generated and sent to the ECS via DMS.

If the operator replies that the product generation request cannot be canceled, this procedure updates the cancellation status in the L1_Prod_Requests table and sends the L1_Prod_Cancel_Stat to the ECS via DMS, indicating that the request will not be canceled.

The prq_stat is returned to indicate successful or unsuccessful processing completion.

NAME: prq_term

TITLE: PRQ Termination

BODY: This module disconnects from the database and closes communication facilities. It returns prq_stat to indicate successful or unsuccessful completion.

5.3.2 Work Order Generator (PWG) Task

This subsection describes the PWG task software.

5.3.2.1 Task Overview

The Work Order Generator task creates the work order for the product request specified in the input parameter. It retrieves the product request options from the database and uses this information to determine which procedure is to be used to process the request. It inserts a new work order into the database. The work order includes details on the scripts and parameter values to be used during processing. The work order state is set to indicate to DMS that the work order is ready for L0R ingest.

5.3.2.2 Initialization

The PWG task initialization consists of connecting to the database and retrieving the product request identifier supplied as an input parameter to the task.

5.3.2.3 Normal Operation

The PWG task generates the work order for the product request that is specified in its input parameters. It first calls pwg_init to initialize the task. The information needed from the product request is retrieved from the database by calling pdb_get_req_info. PWG then calls pdb_det_wo_proc to determine which procedure needs to be used to process this request. Next it calls pdb_gen_wo to generate the work order and store it in the LPGS database. PWG creates the work order directories by calling pwg_gen_wo_dir. The work order state is updated to indicate that the L0R data needs to be staged by invoking xdb_upd_wo_state. Finally, PWG calls xdb_upd_prod_status to update the product request status before calling pwg_term to terminate the task.

5.3.2.4 Error Handling

The Work Order Generator reports any errors encountered and exits gracefully when it is unable to continue processing.

5.3.2.5 Design

This subsection presents the design of the PWG task. Figure 5–4 shows the structure chart for the Work Order Generator task. The module specifications for the PWG task are provided below.

NAME: pwg_main

TITLE: PWG Main

BODY: This module first calls on pwg_init to initialize the PWG task. It then invokes pdb_get_req_info using xxx_prod_req_id to get information for that product request. Using the options supplied by the user in the product request, it then calls on pdb_det_wo_proc to identify the canned procedure that is needed to satisfy the product request. It then invokes pdb_gen_wo to actually generate the work order and store its information in the LPGS database. Next it calls on pwg_gen_wo_dir to generate the root directory and subdirectories in which the input, intermediate and output files to be accessed in processing the work order will reside. Then it first calls on xdb_upd_wo_state to set the state of the work order so that DMS will know that the L0 data for the work order needs to be staged. Next it calls on xdb_upd_prod_status to set the status of the product request to indicate that processing of the request has started. Finally, it invokes pwg_term to terminate the PWG task.

NAME: pwg_init

TITLE: Work Order Generator Initialization

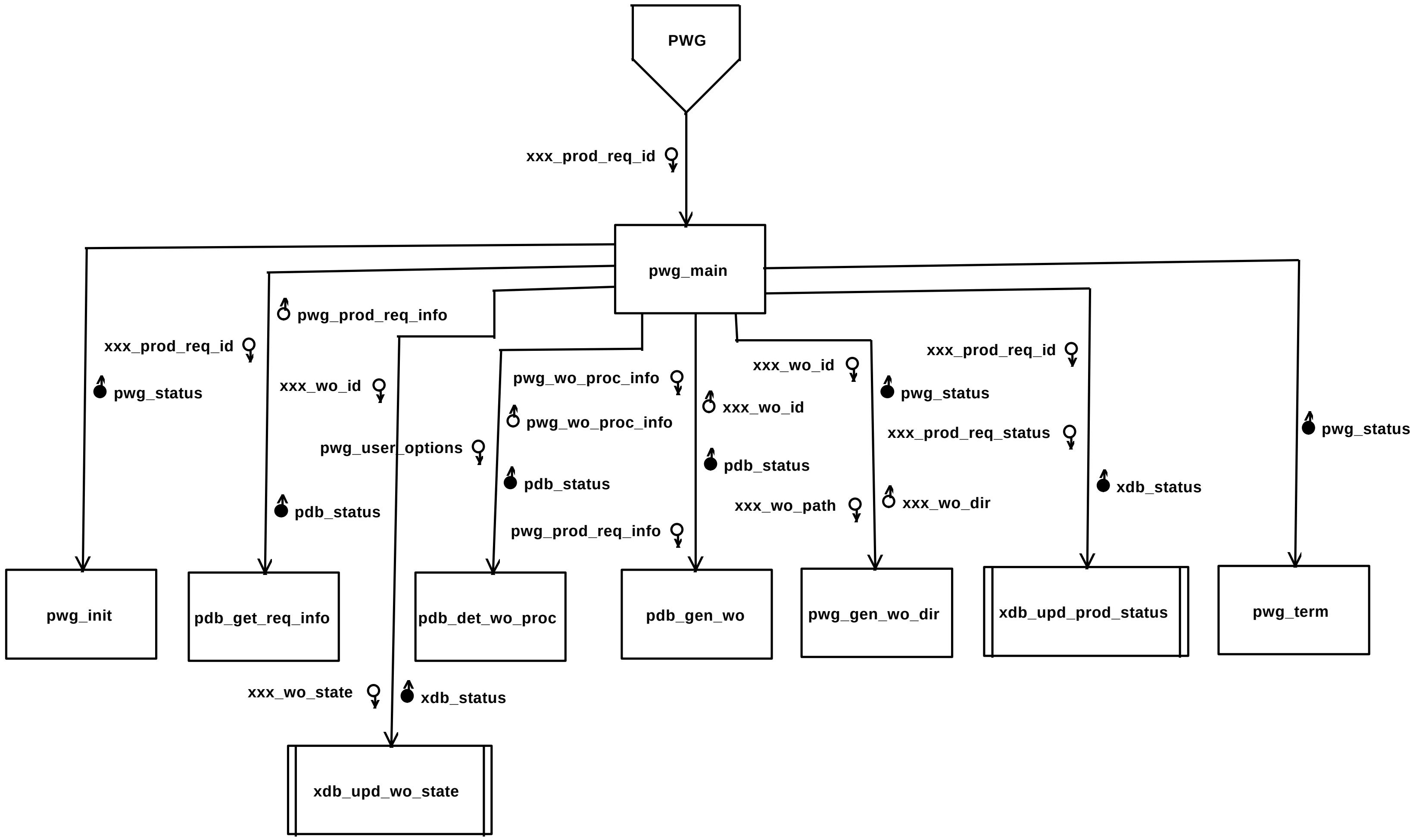

Figure 5–4. Work Order Generator Task Structure Chart

BODY: This module is responsible for initializing the PWG task. Its main functions are to connect the task to the LPGS database and define a signal catcher for catching signals so that they can be reported. It returns pwg_status to indicate whether it was successful or not.

NAME: pdb_get_req_info

TITLE: Get Product Request Information

BODY: This module uses xxx_prod_req_id to retrieve pwg_prod_req_info for a specific product request from the LPGS database. The module returns database status in pdb_status.

NAME: pdb_det_wo_proc

TITLE: Determine Work Order Procedure

BODY: This module uses pwg_user_options to determine which canned procedure to use in processing a particular product request. That information is returned as pwg_wo_proc_info. Database access status is returned in pdb_status.

NAME: pdb_gen_wo

TITLE: Generate Work Order

BODY: This module generates a new work order using pwg_wo_proc_info and pwg_prod_req_info and stores that work order in the LPGS database. In doing so, it identifies the sequence of scripts to be run for the work order and the parameters to be used with each script. When it is done, it returns xxx_wo_id, the unique id of the work order, and pdb_status, which indicates whether the work order was generated successfully or not.

NAME: pwg_gen_wo_dir

TITLE: Generate Work Order Directories

BODY: This module generates the input, intermediate and output directories needed for processing a work order using xxx_wo_id and the default path xxx_wo_path to define the location of the work order's root directory. Errors encountered by the module in generating the directories or accessing the database are returned in pwg_status.

NAME: pwg_term

TITLE: Work Order Generator Termination

BODY: This module is responsible for terminating the PWG task when processing is complete. Its main function is to disconnect from the LPGS database. Errors encountered during termination are returned in pwg_status.

5.3.3 Work Order Scheduler (PWS) Task

This subsection describes the PWS task software.

5.3.3.1 Task Overview

The Work Order Scheduler task is responsible for starting up or resuming work order processing. When there are sufficient disk and CPU resources, it performs a fork and exec of the Work Order Controller task to start execution of the work order.

5.3.3.2 Initialization

The PWS task initialization consists of connecting to the database, establishing a link for intertask communications, and retrieving any setup parameters.

5.3.3.3 Normal Operation

PWS performs in a loop waiting for a termination message. It periodically polls the database to determine whether there is work to do. The polling frequency is controlled by a system parameter. PWS calls pws_init to initialize the task and then calls xxx_get_msg to wait for a message. If a message is not received within a time-out period, PWS wakes up and checks the database to determine whether there is work to be performed. First it calls pws_proc_approved to check for any AAS-initiated work orders that have completed and for which the output has been approved for distribution. Next PWS calls pws_chk_child to determine whether any spawned PWC tasks have completed and to adjust the count of the number of actively executing PWCs appropriately. Then it invokes pws_proc_resumable_wo to continue processing of any work orders that can be resumed. Finally, PWS calls pws_proc_l0ready_wo to start any new work orders if there are sufficient system resources.

PWS calls pws_term when it receives a shutdown message from the LPGS termination task.

5.3.3.4 Error Handling

The Work Order Scheduler reports any errors encountered and exits gracefully when it is unable to continue processing.

5.3.3.5 Design

This subsection presents the design of the PWS task. Figure 5–5 shows the structure chart for the Work Order Scheduler task. The module specifications for the PWS task follow.

NAME: pws_main

TITLE: Work Order Scheduler Main

BODY: This module first calls on pws_init to initialize the PWS task. It then loops until it receives a shutdown message, in which case it invokes pws_term to terminate the PWS task.

While looping it first posts a read for the next message using the library module xxx_get_msg. The module then waits until either a message is returned or a time-out occurs. If a message is returned, it processes it. The only message it currently recognizes is the shutdown message. Hence normal processing for this module occurs only at time-outs.

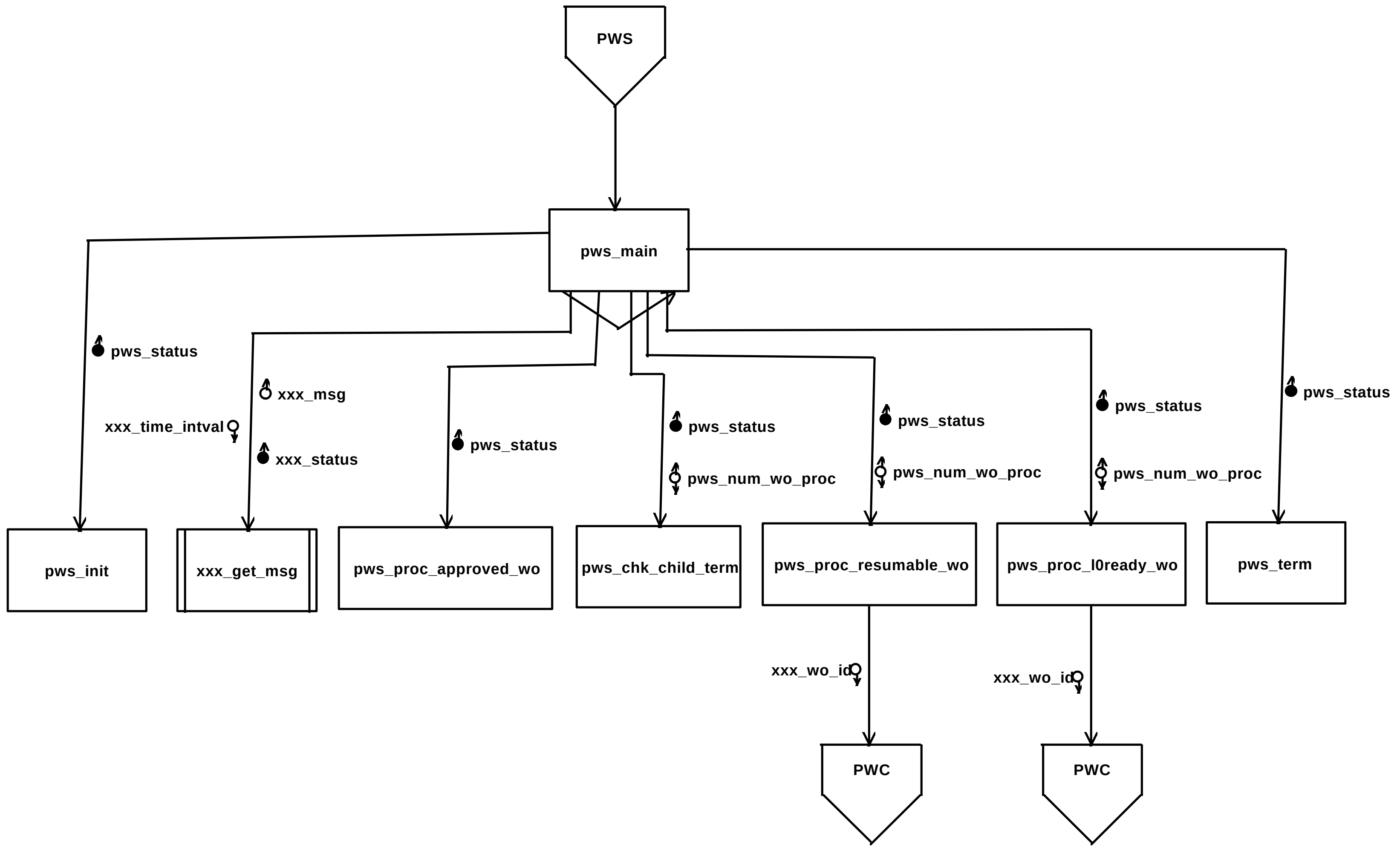

Figure 5–5. Work Order Scheduler Task Structure Chart

When it detects a time-out, pws_main first calls on pws_proc_approved_wo to update the LPGS database so that any completed work orders that were initiated by AAS and whose output the AAS analyst has approved for distribution to the end-user will be transferred to ECS by DMS. It then invokes pws_chk_child_term to determine whether any work order control processes it previously spawned have completed and, if so, to update pws_num_wo_proc, the number of work order control processes still running.

The module then proceeds by calling pws_proc_resumable_wo to continue the processing of any halted or recovered work orders which can now be resumed. Then, if sufficient CPU resources are still available, pws_main calls on pws_proc_l0ready_wo to start the L1 processing of new work orders whose L0 data is available on LPGS and for which sufficient CPU and disk resources are available. It then restarts the loop.

NAME: pws_init

TITLE: Work Order Scheduler Initialization

BODY: This module is responsible for initializing the PWS task. Its main functions are to connect the task to the LPGS database, define a signal catcher for catching signals so that they can be reported, and set up a communication channel so that the task can receive messages from and send messages to other LPGS tasks. It returns pws_status to indicate whether it was successful or not.

NAME: pws_proc_approved_wo

TITLE: Process Approved Work Orders

BODY: This module is responsible for detecting successfully completed work orders whose processing was requested by AAS and whose products were approved for distribution to the end-user. The module looks for all such work orders and, for each one it finds, it updates the status information associated with the work order/product request in the LPGS database and sends L1_Prod_Xfer_Req to DMS to notify it that the L1 product is ready for transfer to ECS. Processing errors encountered by the module are returned in pws_status.

NAME: pws_chk_child_term

TITLE: Check Child Termination

BODY: This module determines whether any work order control processes spawned by the PWS task have terminated and, if so, adjusts pws_num_wo_proc, the total number of work order control processes actually running. Errors encountered by the module are returned in pws_status.

NAME: pws_proc_resumable_wo

TITLE: Process Resumable Work Orders

BODY: This module is responsible for continuing the processing of interrupted work orders whose resumption has been approved by the AAS analyst or operator. The module loops through all such work orders until either pws_num_wo_proc exceeds a prespecified threshold or no more resumable work orders are found.

Within the loop, the module first fetches the next resumable work order based on its priority and creation time. If a work order is found, the module then updates the state of the work order in the LPGS database, invokes an instance of the PWC task as an independent child process, and increments pws_num_wo_proc, the actual number of work order control processes running.

Processing errors encountered by the module are returned in pws_status.

NAME: pws_proc_l0ready_wo

TITLE: Process Ready Work Orders

BODY: This module is responsible for starting the processing of new work orders for which L0 data is available. The module loops through all such work orders until either pws_num_wo_proc exceeds a prespecified threshold, disk space usage as measured by the number of started or completed work orders whose files still reside on LPGS exceeds another threshold, or no more startable work orders are found.

Within the loop, the module first checks whether sufficient disk space is available to support another work order. If sufficient disk space is available, it fetches the next startable work order based on its priority and creation time. If a work order is found, the module then proceeds to check whether a confirmed cancellation request for the product associated with the current work order exists in the LPGS database.

If such a cancellation request exists, the module calls a lower-level module to perform the cancellation and send the L1_Prod_Cancel_Ack message acknowledging the completion of the request to the PRQ task. If no such cancellation request exists, the module updates the state of the work order in the LPGS database, invokes an instance of the PWC task as an independent child process, and increments pws_num_wo_proc, the actual number of work order control processes running.

Processing errors encountered by the module are returned in pws_status.

NAME: pws_term

TITLE: Work Order Scheduler Termination

BODY: This module is responsible for terminating the PWS task when processing is complete. Its main function is to disconnect from the LPGS database and from the communication channel it used to exchange messages with other LPGS tasks. Errors encountered during termination are returned in pws_status.

5.3.4 Work Order Controller (PWC) Task

This subsection describes the PWC task software.

5.3.4.1 Task Overview

The Work Order Controller task is responsible for starting and monitoring the work order scripts for the specified work order. It continues processing scripts until the last script completes successfully, an error occurs, or a halt is encountered. Before starting a new script it checks to see if the work order needs to be canceled. When there are no more scripts to process, PWC sends a message to DMS that a product is ready for transfer to ECS. If an error occurs that requires investigation, PWC notifies the AAS analyst by generating an alert and recording the anomaly in the database.

5.3.4.2 Initialization

The PWC task initialization consists of connecting to the database and retrieving the work order identifier supplied as an input parameter to the task.

5.3.4.3 Normal Operation

PWC begins processing by calling pwc_init to initialize the task. Then it performs a loop until it completes processing of the last script or encounters a halt. Before starting a script, it calls pwc_check_cancel_stat to determine whether the work order needs to be canceled. If it does, PWC performs the cancellation and sends PRQ a message. Otherwise, PWC calls pwc_start_next_script to run the next script. When a script completes it returns a status indicating success, failure or anomaly. PWC calls pwc_proc_success_script_stat, pwc_proc_failed_script_stat, or pwc_proc_anomaly_script_stat, respectively, to process the script completion status. When there are no more scripts or a halt is encountered, pwc_term is called to terminate the task.

5.3.4.4 Error Handling

The Work Order Controller reports any errors encountered and exits gracefully when it is unable to continue processing.

5.3.4.5 Design

This subsection presents the design of the PWC task. Figure 5–6 shows the structure chart for the Work Order Controller task. The module specifications for the PWC task are provided below.

NAME: pwc_main

TITLE: Work Order Controller Main

BODY: This module is invoked when there are more scripts to execute for a work order identified by xxx_wo_id. The pwc_init module is first called to initialize the task. Then a set of commands is executed in a loop until either there are no more scripts for the work order or a halt condition has been encountered.

The loop starts by calling the pwc_check_cancel_stat module to read the L1_Prod_Requests table to check whether the work order needs to be canceled. If the work order needs to be canceled, the work order directories for this work order are marked for deletion, and status information in the L1_Prod_Requests and Work_Orders tables is updated. In addition, the L1_Prod_Cancel_Ack is sent to the Request Processor task, and pwc_term is invoked to end this task.

REVIEW

If the work order does not need to be canceled, this module calls the pwc_start_next_script module to determine which script to run next for this work order

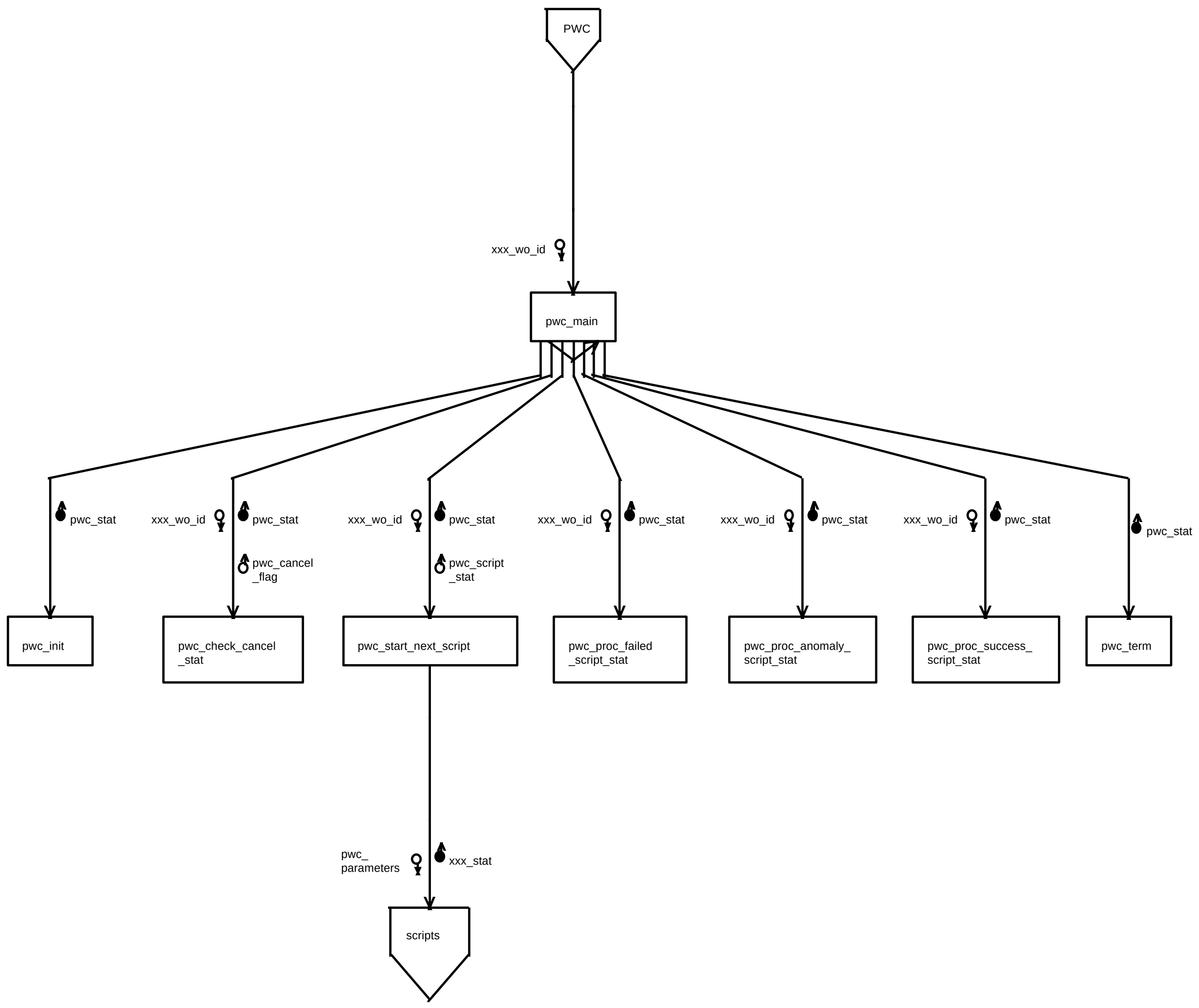

Figure 5–6. Work Order Controller Task Structure Chart

as well as to start the script and wait to receive its completion status. If the completion status indicates that the script failed, the pwc_proc_failed_script_stat module is invoked to process the failure.

If the completion status indicates that the script succeeded, this module invokes the pwc_proc_success_script_stat to process the successful path.

The script could also end with an "anomaly" condition. In this case the pwc_proc_anomaly_script_stat module is invoked to process the anomaly.

If either there are no scripts left for this work order or a halt is encountered, the pwc_term module is invoked to terminate the task.

NAME: pwc_init

TITLE: PWC Initialization

BODY: This module initializes the Work Order Controller task by connecting to the database. The module returns the pwc_stat to indicate successful or unsuccessful completion of the initialization.

NAME: pwc_check_cancel_stat

TITLE: Check Cancel Status

BODY: This module uses the xxx_wo_id to check if the work order needs to be canceled. If the work order needs to be canceled this module marks the directories associated with this work order as deletable. The module also updates the Work_Orders and L1_Prod_Request tables with cancellation data, sends a message to the Request Processor task, and returns the pwc_cancel_flag with a true condition. If the work order has not been canceled, the module returns the pwc_cancel_flag with a false condition.

The pwc_stat is sent to indicate successful or unsuccessful completion of the module.

NAME: pwc_start_next_script

TITLE: Start the Next Script

BODY: This module retrieves the identifier of the next script that is to be run for the work order identified by xxx_wo_id from the Work_Orders table. It forks a task to run the identified script and waits for the script’s completion. Upon completion of the script, it returns the pwc_script_stat to indicate the script’s status along with the pwc_stat which indicates whether this module completed successfully or not.

NAME: pwc_proc_failed_script_stat

TITLE: Process Failed Script Status

BODY: This module is invoked if a script ends with a “failed” status. The module uses the xxx_wo_id to update the status in the Work_Orders table. The module also checks the type of work order that is being processed, normal or AAS, and sends an alert to the appropriate operator to indicate the failed status. It then returns the pwc_status to indicate successful or unsuccessful completion.

NAME: pwc_proc_anomaly_script_stat

TITLE: Process Anomaly Script Status

BODY: This module is invoked if a script ends with an “anomaly” status. It updates the work order status in Work_Orders table using xxx_wo_id, sends an alert to the AAS analyst, and inserts anomaly information into the Anomaly table. It also returns the pwc_stat to indicate successful or unsuccessful completion.

NAME: pwc_proc_success_script_stat

TITLE: Process Successful Script Status

BODY: This module is invoked when a script has ended successfully or when a “halt” condition has been encountered. If a halt is not associated with the script, the script status is updated to indicate the successful completion of the script. If this is the last script for this work order then the work order is updated to show that it is complete, and the L1_Prod_Xfer_Req is sent to DMS to inform it that the L1 product can now be transferred to the ECS.

If this is not the last script for this work order, the script number for this work order is incremented in the Work_Orders table so that the next time a script needs to be run, the next appropriate script will be selected.

If a halt is associated with the script, the work order is updated to indicate a halt for this xxx_wo_id, and an alert is sent to the appropriate operator.

The pwc_stat is returned to indicate successful or unsuccessful completion of the module.

NAME: pwc_term

TITLE: PWC Termination

BODY: This module disconnects from the database and closes communication facilities. It returns pwc_stat indicating successful or unsuccessful completion.